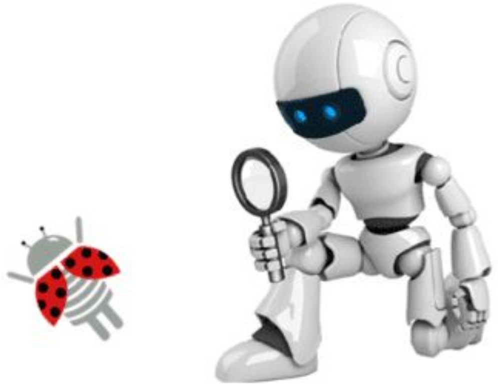
Detox

Detoxification in 5 minutes

What is e2e?

End-to-End (E2E) testing is the practice of running your app on a real device or simulator and interacting with it like a real world user would.

For simplicity, this means that a machine/robot is clicking through your app and checks whether or not a button can be clicked, a text can be typed in the search field or whatever.



Are there any testing frameworks?

Two major E2E testing frameworks exist for React Native projects:

1) Appium: https://github.com/appium/appium

2) Detox: https://github.com/wix/Detox

Description

Appium is a **Black box** testing. It is looking at a system where you don’t know what is inside or how the inside behaves. You only know what you put in and what you expect as an outcome.

Detox is a **Gray box** testing. It is similar to a black box, with the addition that you also have knowledge over the internal behaviour of the system.

Both frameworks are open source, have a decent speed and support at least iOS and Android simulators/emulators.

Comparison

Detox:

- 4 years in industry
- focused on JS
- smaller community
- works in sync with the app
- grey box

- created for React Native
- works faster

Appium:

- 7 years in industry
- focused on different languages
- larger community
- no sync with the app
- black box

- large API capabilities

Installation

First of all you need to install applesimutils:

```
brew tap wix/brew
brew install applesimutils
```

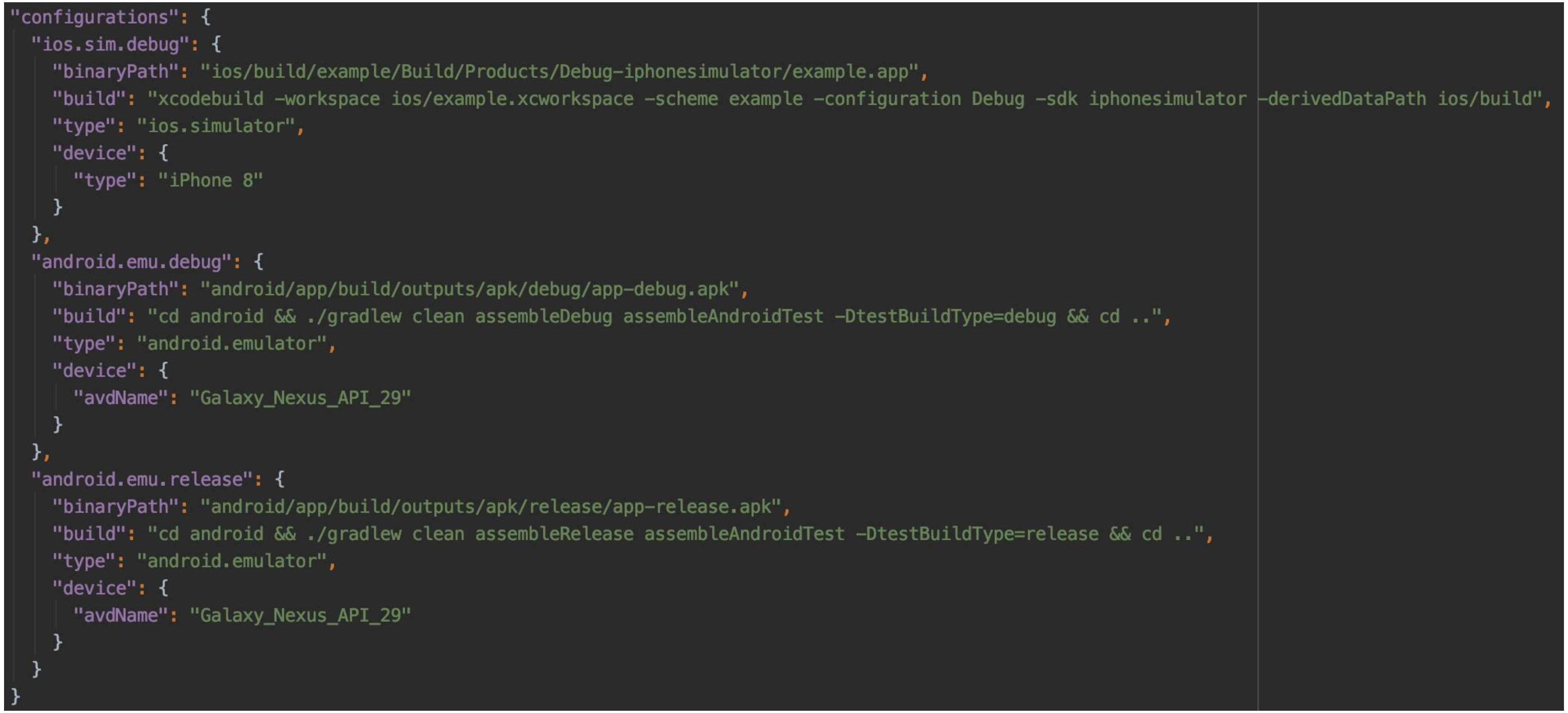
Then install detox-cli:

```
npm install -g detox/cli
```

The last step is:

```
npm install detox --save-dev / yarn add detox -D
```

Configurations



```
"configurations": {
  "ios.sim.debug": {
    "binaryPath": "ios/build/example/Build/Products/Debug-iphonesimulator/example.app",
    "build": "xcodebuild -workspace ios/example.xcworkspace -scheme example -configuration Debug -sdk iphonesimulator -derivedDataPath ios/build",
    "type": "ios.simulator",
    "device": {
      "type": "iPhone 8"
    }
  },
  "android.emu.debug": {
    "binaryPath": "android/app/build/outputs/apk/debug/app-debug.apk",
    "build": "cd android && ./gradlew clean assembleDebug assembleAndroidTest -DtestBuildType=debug && cd ..",
    "type": "android.emulator",
    "device": {
      "avdName": "Galaxy_Nexus_API_29"
    }
  },
  "android.emu.release": {
    "binaryPath": "android/app/build/outputs/apk/release/app-release.apk",
    "build": "cd android && ./gradlew clean assembleRelease assembleAndroidTest -DtestBuildType=release && cd ..",
    "type": "android.emulator",
    "device": {
      "avdName": "Galaxy_Nexus_API_29"
    }
  }
}
```

Running your first test

Assuming you already have Jest or Mocha installed, the next thing to do is:

detox init -r jest(mocha)

This command will generate for you ***e2e*** folder, which includes:

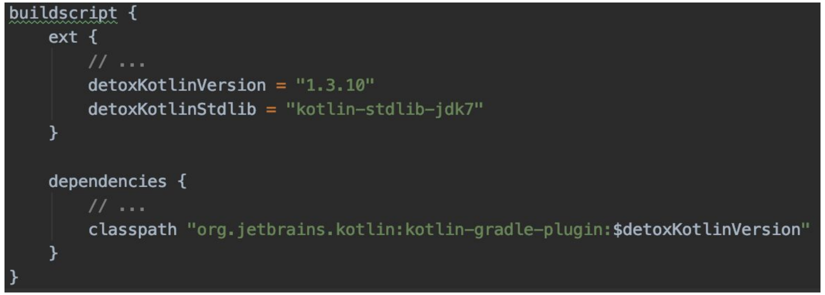
init.js, firstTest.spec.js and config.json(mocha.opts)

Build an app: ***detox build -c ios.sim.debug***

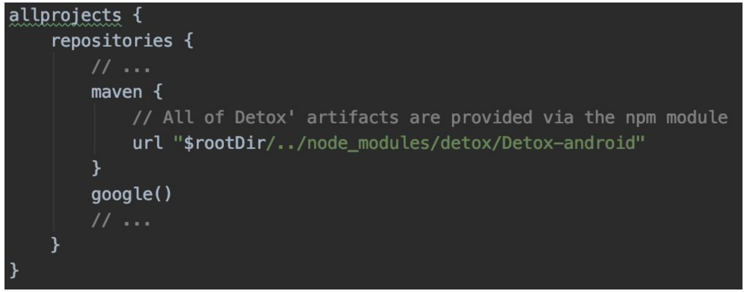
Test it: ***detox test -c ios.sim.debug***

A bit more configuration..

Update ***android/build.gradle***

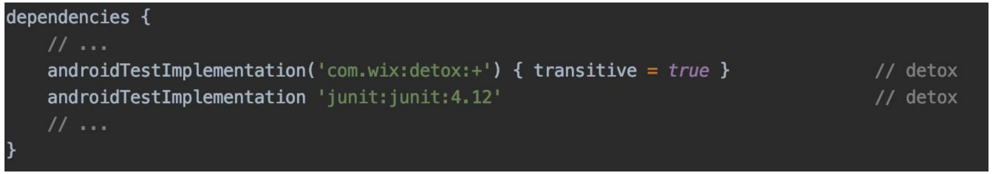


```
buildscript {
    ext {
        // ...
        detoxKotlinVersion = "1.3.10"
        detoxKotlinStdlib = "kotlin-stdlib-jdk7"
    }

    dependencies {
        // ...
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$detoxKotlinVersion"
    }
}
```

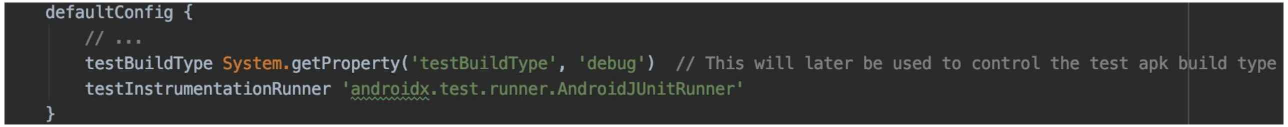


```
allprojects {
    repositories {
        // ...
        maven {
            // All of Detox' artifacts are provided via the npm module
            url "$rootDir/../node_modules/detox/Detox-android"
        }
        google()
        // ...
    }
}
```

Update ***android/app/build.gradle***



```
dependencies {
    // ...
    androidTestImplementation('com.wix:detox:+') { transitive = true }            // detox
    androidTestImplementation 'junit:junit:4.12'                                  // detox
    // ...
}
```



```
    defaultConfig {
        // ...
        testBuildType System.getProperty('testBuildType', 'debug')  // This will later be used to control the test apk build type
        testInstrumentationRunner 'androidx.test.runner.AndroidJUnitRunner'
    }
```

And finally

The last Android configuration part is to create the file:

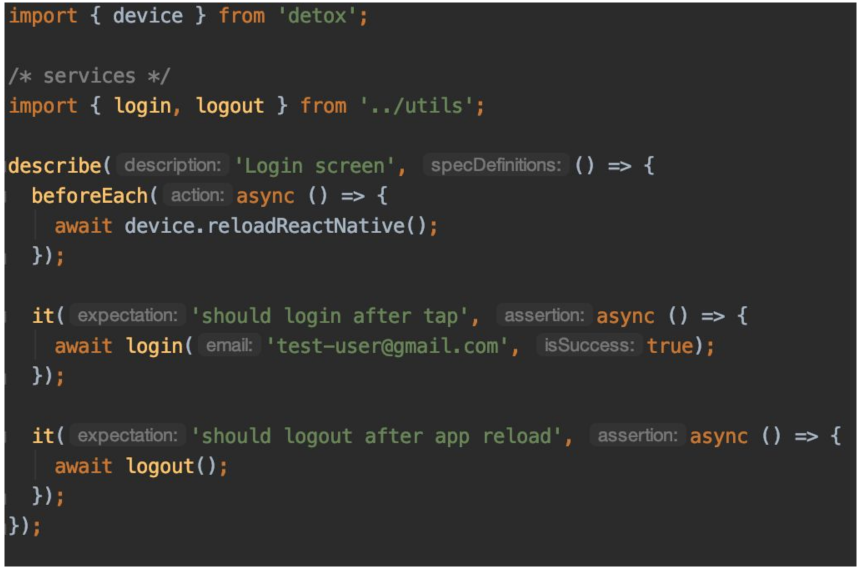
android/app/src/androidTest/java/com/[your.package]/DetoxTest.java

You can simply copy/paste it from there:

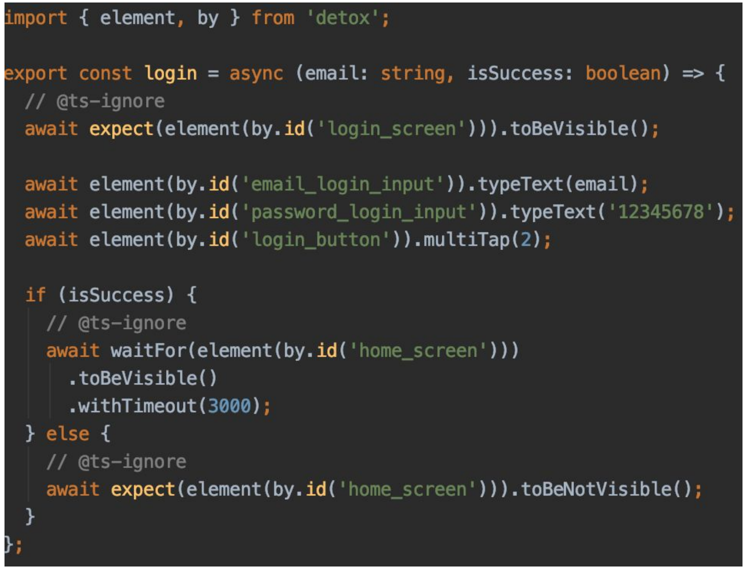
https://github.com/wix/Detox/blob/master/examples/demo-react-native/android/app/src/androidTest/java/com/example/DetoxTest.java

Lets start testing?

The basic login test example:

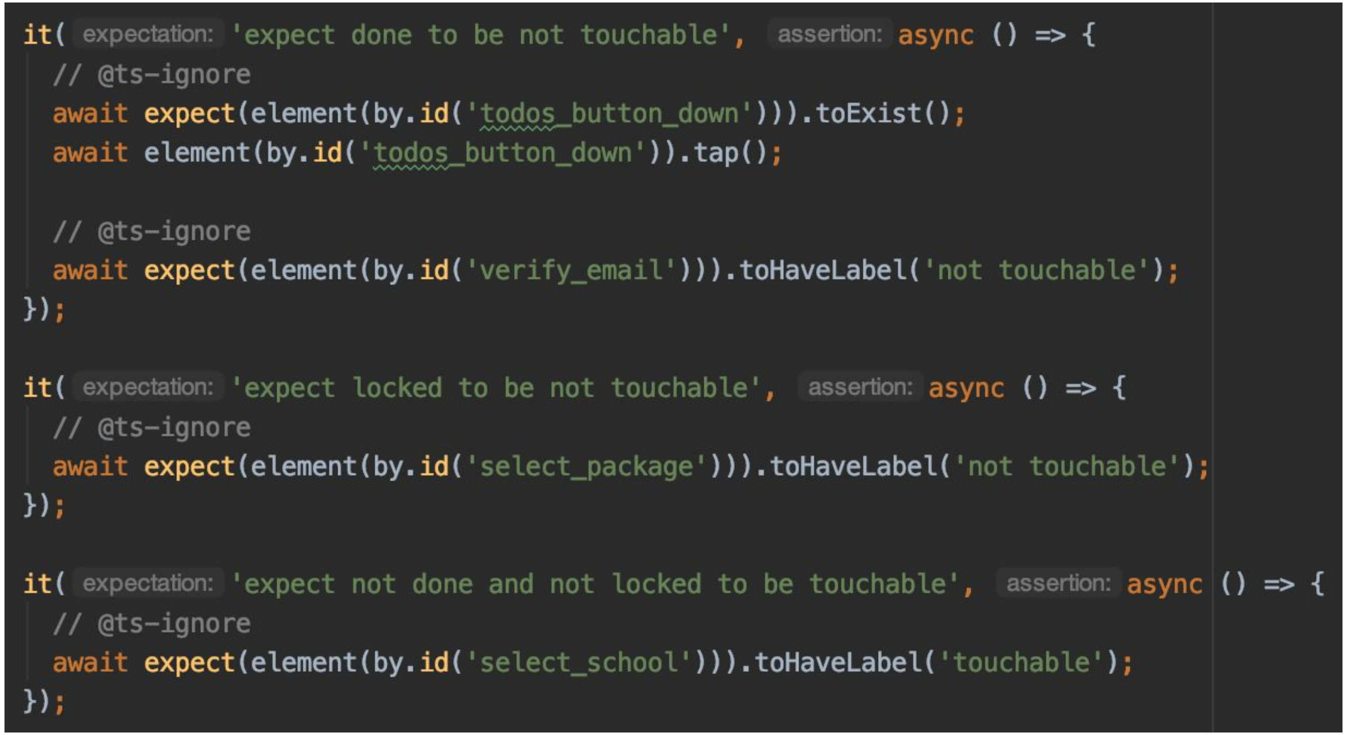


```
import { device } from 'detox';

/* services */
import { login, logout } from '../utils';

describe( description: 'Login screen',  specDefinitions: () => {
  beforeEach( action: async () => {
    await device.reloadReactNative();
  });

  it( expectation: 'should login after tap',  assertion: async () => {
    await login( email: 'test-user@gmail.com',  isSuccess: true);
  });

  it( expectation: 'should logout after app reload',  assertion: async () => {
    await logout();
  });
});
```



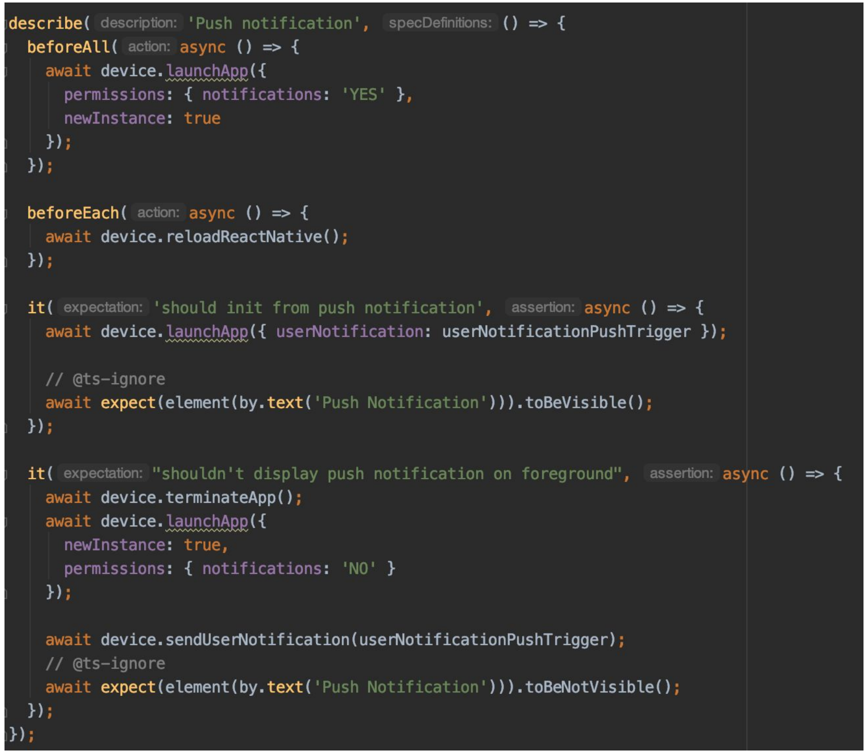
```
import { element, by } from 'detox';

export const login = async (email: string, isSuccess: boolean) => {
  // @ts-ignore
  await expect(element(by.id('login_screen'))).toBeVisible();

  await element(by.id('email_login_input')).typeText(email);
  await element(by.id('password_login_input')).typeText('12345678');
  await element(by.id('login_button')).multiTap(2);

  if (isSuccess) {
    // @ts-ignore
    await waitFor(element(by.id('home_screen')))
      .toBeVisible()
      .withTimeout(3000);
  } else {
    // @ts-ignore
    await expect(element(by.id('home_screen'))).toBeNotVisible();
  }
};
```

Probably 85% of e2e are like

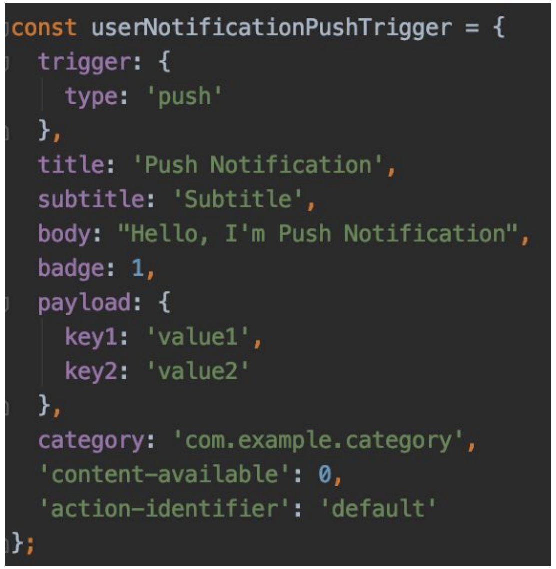


```
it( expectation: 'expect done to be not touchable', assertion: async () => {
  // @ts-ignore
  await expect(element(by.id('todos_button_down'))).toExist();
  await element(by.id('todos_button_down')).tap();

  // @ts-ignore
  await expect(element(by.id('verify_email'))).toHaveLabel('not touchable');
});

it( expectation: 'expect locked to be not touchable', assertion: async () => {
  // @ts-ignore
  await expect(element(by.id('select_package'))).toHaveLabel('not touchable');
});

it( expectation: 'expect not done and not locked to be touchable', assertion: async () => {
  // @ts-ignore
  await expect(element(by.id('select_school'))).toHaveLabel('touchable');
});
```

Testing Push notifications

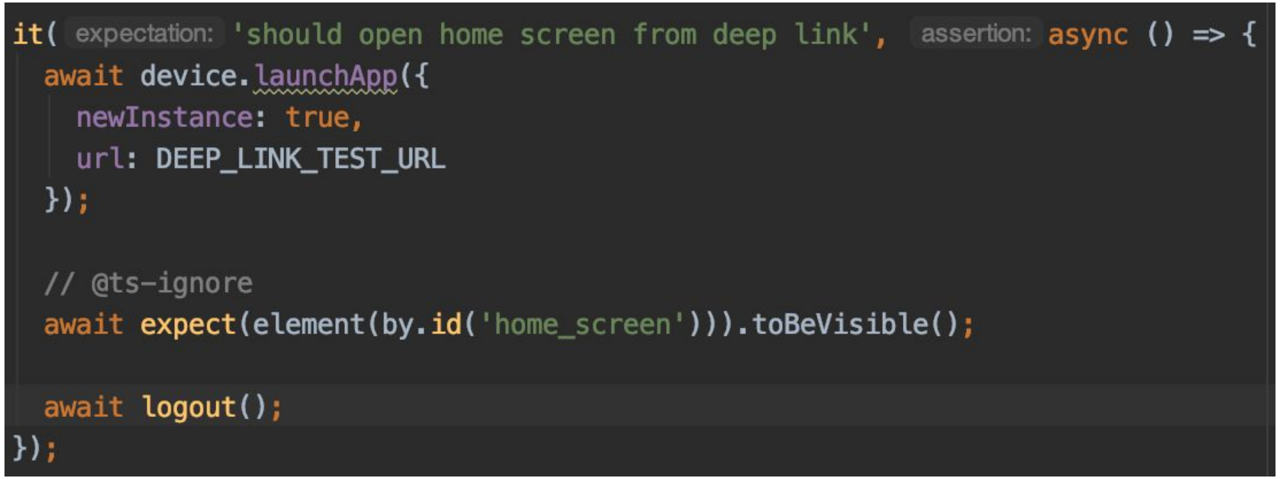


```
describe( description: 'Push notification',  specDefinitions: () => {
  beforeAll( action: async () => {
    await device.launchApp({
      permissions: { notifications: 'YES' },
      newInstance: true
    });
  });

  beforeEach( action: async () => {
    await device.reloadReactNative();
  });

  it( expectation: 'should init from push notification',  assertion: async () => {
    await device.launchApp({ userNotification: userNotificationPushTrigger });

    // @ts-ignore
    await expect(element(by.text('Push Notification'))).toBeVisible();
  });

  it( expectation: "shouldn't display push notification on foreground",  assertion: async () => {
    await device.terminateApp();
    await device.launchApp({
      newInstance: true,
      permissions: { notifications: 'NO' }
    });

    await device.sendUserNotification(userNotificationPushTrigger);
    // @ts-ignore
    await expect(element(by.text('Push Notification'))).toBeNotVisible();
  });
});
```



```
const userNotificationPushTrigger = {
  trigger: {
    type: 'push'
  },
  title: 'Push Notification',
  subtitle: 'Subtitle',
  body: "Hello, I'm Push Notification",
  badge: 1,
  payload: {
    key1: 'value1',
    key2: 'value2'
  },
  category: 'com.example.category',
  'content-available': 0,
  'action-identifier': 'default'
};
```

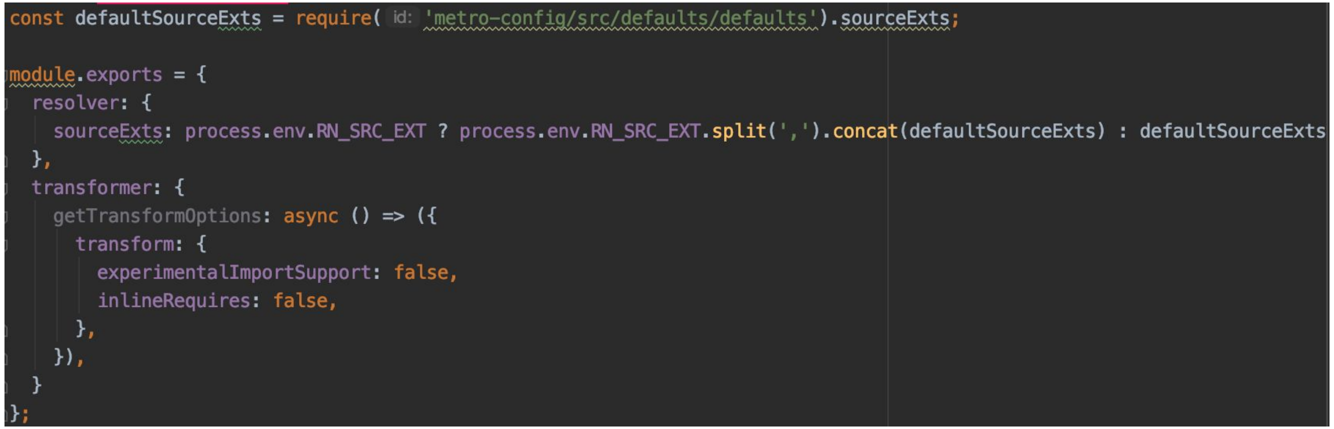
Testing deep linking



```
it( expectation: 'should open home screen from deep link', assertion: async () => {
  await device.launchApp({
    newInstance: true,
    url: DEEP_LINK_TEST_URL
  });

  // @ts-ignore
  await expect(element(by.id('home_screen'))).toBeVisible();

  await logout();
});
```

Mocking configuration

For being able to mock services, your ***metro.config.js*** should looks as follows:



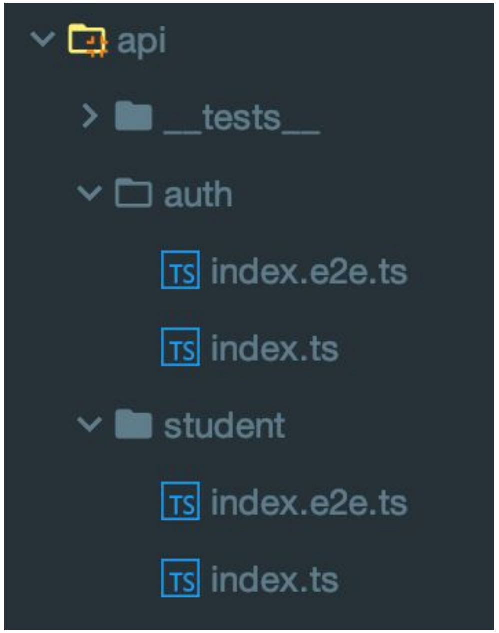
```
const defaultSourceExts = require( id: 'metro-config/src/defaults/defaults').sourceExts;

module.exports = {
  resolver: {
    sourceExts: process.env.RN_SRC_EXT ? process.env.RN_SRC_EXT.split(',').concat(defaultSourceExts) : defaultSourceExts
  },
  transformer: {
    getTransformOptions: async () => ({
      transform: {
        experimentalImportSupport: false,
        inlineRequires: false,
      },
    }),
  }
};
```

Then, instead of regular ***react-native start*** it becomes ***RN_SRC_EXT=e2e.ts react-native start --reset-cache***

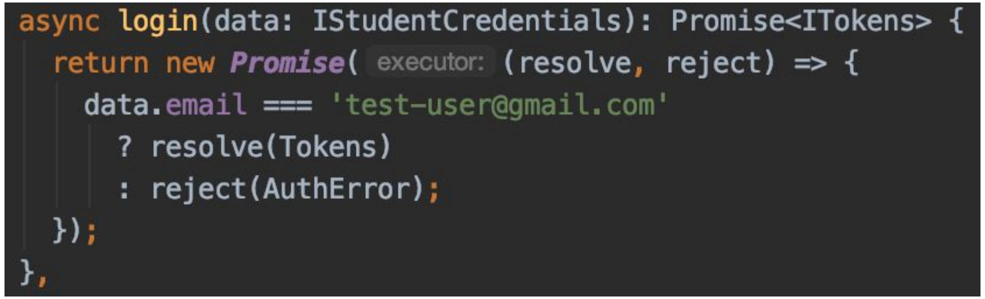
But you’ll need it only before running e2e

Mocking API and other..

After the following changes your api structure will looks like


```
api
  __tests__
  auth
    index.e2e.ts
    index.ts
  student
    index.e2e.ts
    index.ts
```

As an example, login func is:


```
async login(data: IStudentCredentials): Promise<ITokens> {
  return new Promise( executor: (resolve, reject) => {
    data.email === 'test-user@gmail.com'
      ? resolve(Tokens)
      : reject(AuthError);
  });
},
```

Something to note

Thank you ;)

Contacts

```
Twitter:    @stenzets
Telergam:   @stenzets
Facebook:   facebook.com/groups/react.native.belarus/
```

IDEASOFT